Luis Mena Rojas

Resumen Ejecutivo1 Data Formatting & Normalization

Data formatting

Se refiere al proceso de ajustar los datos a un formato específico, lo que facilita su interpretación y análisis. Esto puede incluir acciones como cambiar tipos de datos, ajustar fechas y horas, estandarizar cadenas de texto.

Normalización

puede ser esencial al preparar datos numéricos que varían en diferentes escalas, como precios y temperaturas, para que su variación no afecte desproporcionadamente los resultados.

Para ambos

Los conceptos ayudan a minimizar errores y optimizar el rendimiento de los modelos de análisis, especialmente en Python, donde se dispone de herramientas como pandas y numpy para llevar a cabo estas tareas.

Se procede a la carga de librerías, se invoca a la librería Pandas, la cual permite obtener información de archivos como CSV, Excel, SQL, HTML,

Cargar e integrar datos

Para la carga de datos de los archivos del caso de estudio se empleó la línea de código “df=pd.read_excel('bienes_raices.xlsx')”, donde df corresponde a la variable que obtiene los datos de un archivo de Excel con nombre de “bienes_raices.xlsx”, variable de tipo DataFrame. Así también, se puede utilizar el comando “print(df.head())”, con la finalidad de imprimir las primeras 5 filas, para asegurar que los datos han sido cargados.

Eliminar columnas duplicadas

e inservibles Antes de la eliminación de filas se debe realizar un análisis de las columnas que pueden poseer los mismos datos

Eliminar filas duplicadas

La línea de código utilizada para eliminar filas duplicadas es “df=df.drop_duplicates()”, adicionalmente para reiniciar el índice del DataFrame

Eliminar filas Inservibles

Definir el porcentaje de columnas válidas deseado 80% “porcentaje_validas=0.8”. Se cuenta las columnas válidas para cada fila “num_validas=df.notnull().sum(axis=1)”

En Python se usa este comando, “df=df.drop(‘Nombre_Columna’,axis=1)”

Calcular el porcentaje de columnas

válidas para cada fila “porc_validas=num_validas/df.shape[1]”. Seleccionar las filas que tengan menos porcentaje que lo permitido “filas_validas=porc_validas>=porcentaje_validas”.

Eliminar las filas que no cumplan con este criterio

“df=df[filas_validas]”

Reiniciar el índice del dataframe “df=df.reset_index(drop=True)”.

Verificar errores en datos numéricas

Encontrando la ubicación

“mask=df['Precio'].astype(str).str.contains('usd',case=False)” del problema, Reemplazando en toda la columna donde existen problemas relacionados a usd, “df.loc[mask,'Precio']=df.loc[mask,'Precio'].str.replace('usd','')”. Aquí se reemplaza los valores que contengan espacios en blanco, por ningún valor, mask=df['Precio'].astype(str).str.contains(' ',case=False)

Precio

Posteriormente ya es posible formatear la variable inicialmente de tipo object, a un valor numérico de float “df['Precio']=df['Precio'].astype(float)”.

Realizar el tratamiento de datos vacíos

“imputer_num=SimpleImputer(strategy='mean')
imputer_cat=SimpleImputer(strategy='most_frequent')
df['Precio']=imputer_num.fit_transform(df[['Precio']]) df['Tipo de propiedad']=imputer_cat.fit_transform(df[['Tipo de propiedad']])”

Exportar datos

se utiliza el comando df.to_excel('base_limpia.xlsx',index=False), donde “base_limpia.xlsx” representa el nombre del nuevo archivo que contiene la base de datos en formato Excel y que será utilizada posteriormente para los diferentes análisis.